

A CMA-ES Super-fit Scheme for the Re-sampled Inheritance Search

Fabio Caraffini ^{*} [†], Giovanni Iacca[‡], Ferrante Neri ^{*} [†], Lorenzo Picinali^{*}, Ernesto Mininno[†]

^{*} Centre for Computational Intelligence, School of Computer Science and Informatics, De Montfort University,
The Gateway, Leicester LE1 9BH, United Kingdom
Email: fabio.caraffini@email.dmu.ac.uk, fneri@dmu.ac.uk, and lpicinali@dmu.ac.uk

[†] Department of Mathematical Information Technology,
P.O. Box 35 (Agora), 40014 University of Jyväskylä, Finland
Email: fabio.caraffini@jyu.fi, ferrante.neri@jyu.fi, and ernesto.mininno@jyu.fi

[‡]INCAS³ - Innovation Centre for Advanced Sensors and Sensor Systems
P.O. Box 797, 9400 AT Assen, The Netherlands
Email: giovanniiacca@incas3.eu

Abstract—The super-fit scheme, consisting of injecting an individual with high fitness into the initial population of an algorithm, has shown to be a simple and effective way to enhance the algorithmic performance of the population-based algorithm. Whether the super-fit individual is based on some prior knowledge on the optimization problem or is derived from an initial step of pre-processing, e.g. a local search, this mechanism has been applied successfully in various examples of evolutionary and swarm intelligence algorithms. This paper presents an unconventional application of this super-fit scheme, where the super-fit individual is obtained by means of the Covariance Adaptation Matrix Evolution Strategy (CMA-ES), and fed to a single solution local search which perturbs iteratively each variable. Thus, compared to other super-fit schemes, the roles of super-fit individual generator and global optimizer are switched. To prevent premature convergence, the local search employs a re-sampling mechanism which inherits parts of the best individual while randomly sampling the remaining variables. We refer to such local search as Re-sampled Inheritance Search (RIS). Tested on the CEC 2013 optimization benchmark, the proposed algorithm, named CMA-ES-RIS, displays a respectable performance and a good balance between exploration and exploitation, resulting into a versatile and robust optimization tool.

I. INTRODUCTION

Despite the important theoretical framework of the no free lunch theorem [1], which proved that a “universal” optimizer able to find the optimum on all possible fitness functions does not exist, or, in other words, that the average performance of any two algorithms on all possible fitness functions is equivalent, researchers all over the world are still putting a strong effort in designing robust and versatile algorithms for tackling diverse problems. The idea is that it is worthwhile, from both a conceptual and practical point of view, to study novel algorithmic structures that, if properly tuned, can guarantee a robust and reliable performance at least on a broad range of optimization problems. With this aim in mind, the research community developed, in recent years, increasingly complex algorithms made up, in the Memetic Computing fashion, of multiple components (memes) and/or using sophisticated

learning techniques able to adapt the algorithmic behaviour to the features of the fitness landscape, e.g. multi-modality, ill-conditioning, non-separability. Some examples of modern algorithms of this kind can be found in [2]–[6].

Such modern algorithmic structures, on the other hand, are often characterized by an excessive computational overhead, both in terms of memory and CPU operations, or by a high number of parameters which require a fine-tuning on the problem at hand. Thus, their applicability is sometimes limited in practical contexts. In addition to that, as shown in [7] and [8], minimalistic memetic approaches featuring two or three simple components, if properly coordinated, show in many cases at least the same performance as overwhelmingly complex algorithmic structures, yet requiring a much lower computational overhead. In [7], this finding was explained in the light of the Ockham’s Razor, i.e. applying the principle of parsimony to Memetic Computing, and, in general, to the algorithmic design process: minimal structures composed of few simple memes should be preferred to complex ones.

In this paper, we propose to combine the law of parsimony with the super-fit logic by enhancing upon the performance of a multi-start local search. The latter, namely Re-sampled Inheritance Search (RIS), is a simple single-solution algorithm composed of two memes, exploitative local search and a re-sampling mechanism, respectively. The idea of super-fit individual, introduced in the context of Differential Evolution in [9] and [10], consists of “drugging” the initial population by injecting one individual (the so called super-fit) resulting from an exploitative search. The investment of an initial budget for performing an exploitative search pays off by a much high performance of the population-based framework, see [9] and [10]. In Differential Evolution frameworks, the super-fit individual appears to lead the search and attract the other candidate individuals towards the most promising area of the search space, similarly to the global best particle in swarm intelligence. This injection of a super-fit individual into the initial population has the clear effect of promoting exploitation

over exploration. However, in [9] it is shown that, since Differential Evolution naturally maintains a certain degree of population diversity during the optimization process, the super-fit scheme allows a good balance between exploitation and exploration without excessively biasing the search but rather gently guiding it.

In the approach we propose here, instead of using a local search for super-fit individual generation as in [9] and [10], we use a population-based algorithm, namely the Covariance Adaptation Matrix Evolution Strategy (CMA-ES). Different from a local search, CMA-ES has a lower chance of being trapped into a local optimum, providing a super-fit individual which is likely close to the global optimum. After a limited number of fitness evaluations, the super-fit individual found by CMA-ES is then passed to RIS. The latter is supposed to improve upon it and, in case of premature convergence, restart the search inheriting some information from the best individual. The resulting algorithm, named CMA-ES-RIS, is thus characterized by a proper combination of exploration and exploitation. Most of all, its simple structure allows to distinguish clearly the contribution of each component to the overall algorithmic performance.

The remainder of this paper is organized as follows. Section II describes the CMA-ES-RIS structure and its components. Section III presents the numerical results obtained with the CMA-ES-RIS algorithm on the CEC 2013 real-parameter optimization benchmark in three different dimensionalities, namely 10, 30 and 50 and some considerations on its computational complexity. Finally, Section IV concludes this work and suggests a possible follow-up of this research.

II. THE PROPOSED APPROACH

Without loss of generality, in the following we refer to the problem of minimizing an objective function $f(\mathbf{x})$, where the candidate solution $\mathbf{x}$ is a vector of n design variables (or genes) in a decision space $\mathbf{D}$. We indicate with $x[i]$ the i^{th} element of the vector $\mathbf{x}$. The proposed CMA-ES-RIS algorithm attempts to solve this problem at first generating a super-fit individual, and then refining the search to improve upon it. The two stages of the algorithm are executed sequentially by two different components, respectively CMA-ES and RIS, described in the next subsections. The general structure of CMA-ES-RIS is illustrated at the end of this section.

A. Covariance Adaptation Matrix Evolution Strategy (CMA-ES)

The Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [11], [12], [13] is a powerful variant of classic Evolution Strategies (ES), see [14] and [15], which makes use of distribution model of the population. More specifically, at each step of the algorithm λ new solutions are sampled from a multivariate normal distribution whose mean and covariance matrix are adaptively updated from a weighted sum of the best μ solutions in the population. The loop goes on until a stop condition is met. It is interesting to note that, compared to classic evolution strategies employing the (μ, λ) or $(\mu + \lambda)$ survivor selection schemes, CMA-ES is said to employ a $(\mu/\mu_w, \lambda)$ strategy, where μ_w represent the weights on the best μ individuals used to update the distribution model. A detailed

```

initialize covariance matrix  $\mathbf{C} = \mathbf{I}$ , and step-size  $\sigma$ 
initialize the mean vector  $\mathbf{m}$  with a random sample in  $\mathbf{D}$ 
while CMA-ES stop criterion is not met do
    (activate Covariance Matrix Repairing method, if needed)
    sample  $\lambda$  new individuals from distribution  $\mathcal{N}(\mathbf{m}, \sigma^2 \mathbf{C})$ 
    evaluate individuals and sort them based on their fitness
    update  $\mathbf{m}$  based on a weighted sum of the best  $\mu$  individuals
    update covariance matrix  $\mathbf{C}$  and step-size  $\sigma$ 
end while
return best individual  $\mathbf{x}_e$ 

```

Fig. 1. Pseudo-code of CMA-ES

description of CMA-ES is out of the scope of this paper, while interested readers can find all relevant information in the vast literature on this algorithm. We report instead, for the sake of completeness, a simplified pseudo-code illustrating the main steps of CMA-ES, see Fig. 1.

Due to its solid theoretical background, its simple parameter setting, and its invariance to several transformations of the fitness function, CMA-ES is considered one of the state-of-the-art stochastic real-parameter optimization algorithms. In the last decade, it has been used as building block in a number of complex modern algorithms, either extending its original structure, introducing e.g. a restart scheme and an increasing population size as in [16]; hybridising it with other meta-heuristics, such as DE, see the DCMA-EA algorithm proposed in [17]; using it as an intense local search in a memetic structure, as proposed in [2]; or even simplifying its scheme as in [18], where a computationally efficient (1+1)-CMA-ES with Cholesky update for the covariance matrix is presented. Here we follow this research trend, using CMA-ES as super-fit individual generator for the Re-sampling Inheritance Search described in the next subsection.

As shown in Fig. 4, we here refer to the super-fit individual generated by CMA-ES as to $\mathbf{x}_e$ (“elite”). As a final remark, it is important to note that in this work we introduced into the original scheme of CMA-ES, the Covariance Matrix Repairing (CMR) method devised in [19] to fix ill-posed covariance matrices. This mechanism, activated whenever the distribution model has to be used to sample new solutions, works as follows: at the beginning, a correction factor $K = 1$ is initialized. Then, for a fixed number of iterations, first the eigenvalues of $\mathbf{C}$ are calculated: if all of them are positive, the loop is stopped as this condition ensures that $\mathbf{C}$ is well-posed; otherwise, if at least one of the eigenvalues is negative, i.e. $\mathbf{C}$ is ill-posed, the CMR update rule $\mathbf{C} = \mathbf{C} + |\gamma| \cdot K \cdot \mathbf{I}$ is applied, where γ is the minimum (negative) eigenvalue of $\mathbf{C}$. The factor K is subsequently multiplied by a factor $\Delta = 1.5$ and the loop goes on until a maximum number of iterations is reached. In [19], it is shown that 5 iterations are generally enough to guarantee the numerical stability of $\mathbf{C}$, as we also confirmed during our numerical experiments.

B. Re-sampling Inheritance Search (RIS)

The Re-sampling Inheritance Search (RIS) is an extremely simple single-solution algorithm for numerical optimization designed on the basis of the Ockham Razor’s principle in MC, see [7]. The RIS algorithm makes use of only two operators which perturb, alternatively, a single solution. In the CMA-ES-RIS scheme, the initial solution is obviously the super-fit individual $\mathbf{x}_e$ generated by the CMA-ES during the previous

```

input:  $\mathbf{x}_e$ 
generate a random solution  $\mathbf{x}_t$  within  $\mathbf{D}$ 
generate  $i = \text{round}(n \cdot \text{rand}(0, 1))$ 
compute  $Cr$  according to eq. (1)
 $x_t[i] = x_e[i]$ 
 $k = 1$ 
while  $\text{rand}(0, 1) \leq Cr$  and  $k < n$  do
   $x_t[i] = x_e[i]$ 
   $i = i + 1$ 
  if  $i == n$  then
     $i = 1$ 
  end if
   $k = k + 1$ 
end while
if  $f(\mathbf{x}_t) \leq f(\mathbf{x}_e)$  then
   $\mathbf{x}_e = \mathbf{x}_t$ 
end if
return  $\mathbf{x}_t, \mathbf{x}_e$ 

```

Fig. 2. Pseudo-code of re-sampling with inheritance

stage. Then, the following two operators are applied to perturb $\mathbf{x}_e$.

1) *Re-sampling with inheritance*: Similar to the exponential crossover used in Differential Evolution [20], this operator modifies the current solution $\mathbf{x}_e$ in order to generate a perturbed solution $\mathbf{x}_t$ (“trial”). With reference to Fig. 2, the operator works as follows: at the beginning, the trial solution is randomly sampled within the decision space $\mathbf{D}$. Then, one gene of $\mathbf{x}_e$ is randomly selected and copied into the corresponding gene of $\mathbf{x}_t$, to ensure that at least one gene of the current solution is inherited by the new individual. Starting from this gene, a copy process of the genes of $\mathbf{x}_e$ into the corresponding genes of $\mathbf{x}_t$ is continued until $r \leq Cr$, where r is a random number in $\mathcal{U}(0, 1)$ – generated for each gene – and Cr (crossover rate) is a parameter affecting the number of transferred genes. As shown in [21], instead of setting Cr explicitly, we rather set the inheritance factor $\alpha_e \approx n_e/n$, where n_e is the expectation of the number of genes copied from $\mathbf{x}_e$ into $\mathbf{x}_t$ in addition to the first gene, which is deterministically copied. Given that the probability that n_e genes are copied is $Cr^{n_e} = Cr^{n\alpha_e}$, it can be easily shown that, in order to achieve that approximately n_e genes are copied into the offspring with probability 0.5, the crossover rate can be set as:

$$Cr = \frac{1}{n\alpha_e/2}. \quad (1)$$

Thus Cr can be easily derived from the problem dimension n and the inheritance factor α_e , making its effect independent on problem dimension. As soon as $r > Cr$, the process is interrupted and the remaining genes of $\mathbf{x}_t$ remained unchanged. The copy process is applied as for a cyclic buffer, i.e. when the n^{th} gene is reached, the next to be copied is the first one.

Once the trial solution $\mathbf{x}_t$ has been generated, its fitness is computed and compared with that of $\mathbf{x}_e$. In case of improvement, $\mathbf{x}_t$ replaces $\mathbf{x}_e$; otherwise, the elite $\mathbf{x}_e$ is not modified. In both cases, the trial solution $\mathbf{x}_t$ is passed to the next operator, i.e. the exploitative local search, which will use it as a starting point. Thus, this mechanism can be seen as a “smart” restart for the exploitative local search, i.e. a non-destructive re-sampling which preserves parts of the genes of the current best individual in order to reuse the achievements obtained so far by the algorithm.

```

input:  $\mathbf{x}_t, \mathbf{x}_e$ 
 $\rho = \delta \cdot (ub - lb)$ 
while  $\rho < \varepsilon$  do
   $\mathbf{x}_s = \mathbf{x}_t$ 
  for  $i = 1 : n$  do
     $x_s[i] = x_t[i] - \rho$ 
    if  $f(\mathbf{x}_s) \leq f(\mathbf{x}_t)$  then
       $\mathbf{x}_t = \mathbf{x}_s$ 
    else
       $x_s[i] = x_t[i] + \frac{\rho}{2}$ 
      if  $f(\mathbf{x}_s) \leq f(\mathbf{x}_t)$  then
         $\mathbf{x}_t = \mathbf{x}_s$ 
      end if
    end if
  end for
  if  $f(\mathbf{x}_t) \leq f(\mathbf{x}_e)$  then
     $\mathbf{x}_e = \mathbf{x}_t$ 
  else
     $\rho = \frac{\rho}{2}$ 
  end if
end while
return  $\mathbf{x}_e$ 

```

Fig. 3. Pseudo-code of the exploitative local search

2) *Exploitative local search*: The exploitative local search algorithm belongs to the family of hill-descend algorithms; more specifically, it employs the search logic of one searcher proposed in [22], which attempts to improve upon a given solution perturbing separately, on both directions, each of its design variables. With reference to Fig. 3, this operator performs, iteratively, the following operations. At the beginning, the trial solution $\mathbf{x}_t$ generated by the first operator is copied into a temporary solution $\mathbf{x}_s$. The, for each i^{th} variable, the solution $\mathbf{x}_s$ is perturbed by computing:

$$x_s[i] = x_t[i] - \rho, \quad (2)$$

where ρ is the search radius, initialized as fraction δ of the search space $\mathbf{D}$. I.e., $\rho = \delta \cdot (ub - lb)$, where ub and lb are respectively the upper and lower bound of $\mathbf{D}$. If the newly perturbed solution $\mathbf{x}_s$ outperforms the initial trial solution $\mathbf{x}_t$, the trial solution $\mathbf{x}_t$ is replaced by $\mathbf{x}_s$, otherwise a perturbation in the opposite direction is performed, of width $\rho/2$:

$$x_s[i] = x_t[i] + \frac{\rho}{2} \quad (3)$$

and again $\mathbf{x}_t$ is replaced by $\mathbf{x}_s$ in case of improvement. After all the variables have been perturbed in both directions, the elite $\mathbf{x}_e$ is replaced by $\mathbf{x}_t$ if it is outperformed by it. If the elite is not updated, the radius ρ is halved further. The search is then repeated again, for all the design variables, within the new search radius. This process is continued until the radius ρ becomes smaller than a fixed threshold ε .

C. General structure of CMA-ES-RIS

Fig. 4 displays the global structure of CMA-ES-RIS, as serial activation of the components described before. The coordination of the components is rather straightforward. At the beginning of the algorithm, for a given fraction $r_{CMA-ES} \times T$ of the total computational budget T , CMA-ES is executed to generate a super-fit individual $\mathbf{x}_e$. This will be the best individual in the final population processed by CMA-ES. The super-fit individual is then passed to the RIS part of the algorithm, which is continued for the remaining budget $(1 - r_{CMA-ES}) \times T$. During each step of this second stage, first the re-sampling with inheritance processes the current elite

$\mathbf{x}_e$ and generates a trial solution $\mathbf{x}_t$, which is then processed by the exploitative local search. At the end of the local search, the (possibly improved) elite $\mathbf{x}_e$ is passed to the next iteration of RIS, and the loop goes on until the budget T is consumed.

Each component has a clear role in the algorithmic structure and gives a different contribution to the overall performance. As said before, the CMA-ES plays the role of super-fit individual generator: compared to other algorithms, CMA-ES efficiently exploits the parameter linkages, thanks to the covariance matrix, thus converging quickly – and reliably – towards a promising area of the search space. This area represents the starting point for the second stage, i.e. the RIS algorithm, a simple multi-start local search which, in the fashion of Memetic Computing, combines an explorative component (the re-sampling mechanism) with an exploitative local search. A crucial element of RIS is the exchange of information between the two components, in the form of a partial transmission of genes from the elite to the perturbed solution, which balances exploration and exploitation in a simple but apparently effective way.

While super-fit individual generated by CMA-ES is processed by the RIS a double action is taken. The exploitative local search attempts to quickly improve upon the algorithmic performance while the re-sampling mechanism allows an enhancement of the global search features and thus that the solution is trapped within a local optimum. The inheritance mechanism integrated within the re-sampling helps to make the search excessively exploratory and thus randomly search throughout the decision space. Finally, it must be observed that the exploitative local search has a very different search rule with respect to that of CMA-ES. While the first perturbs the solution by moving along the axes, the second follows the shape of the basin of attraction thus performing diagonal moves. The proposed CMA-ES-RIS is supposed to harmonically combine these features and offer a robust performance over a diverse array of optimization problems, including separable and non-separable problems.

From a complexity point of view, it has been shown [13] that, due to the covariance matrix update, the time complexity of the CMA-ES is quadratic. On the other hand, the re-sampling with inheritance performs, on average, $0.5 \cdot n + 1$ copies from the elite to the best, being n the problem dimensionality. In the worst case (which is however rather improbable), it performs n copies. Thus its complexity is $\mathcal{O}(n)$. As for the exploitative local search, it performs, at each step, $2 \cdot n$ perturbations. Therefore also its asymptotic complexity is linear. We can conclude then that the complexity of RIS is $\mathcal{O}(n)$, while the global complexity of CMA-ES-RIS (see below) is between $\mathcal{O}(n)$ and $\mathcal{O}(n^2)$, depending on the value of r_{CMA-ES} .

As a final remark, we note that CMA-ES-RIS makes use of a toroidal handling of the search space bounds: in other words, if $x[i]$ exceeds its upper bound $ub[i]$ by a quantity ζ ($x[i] = ub[i] + \zeta$), the replacement $x[i] = lb[i] + \zeta$ occurs. A specular mechanism is applied on the lower bound.

III. NUMERICAL RESULTS

Following the rules of the CEC 2013 Special Session on Real-Parameter Optimization, we tested the proposed approach

```

CMA-ES stop criterion:  $r_{CMA-ES} \times T$ 
execute CMA-ES as in Fig. 1
RIS stop criterion:  $(1 - r_{CMA-ES}) \times T$ 
while RIS stop criterion is not met do
    execute re-sampling with inheritance as in Fig. 2
    execute exploitative local search as in Fig. 3
end while
return best individual  $\mathbf{x}_e$ 

```

Fig. 4. Pseudo-code of CMA-ES-RIS

on the 28 minimization problems defined in [23]. The whole benchmark was tested in $D = 10$, $D = 30$ and $D = 50$ dimensions, and for each test function and problem size the CMA-ES-RIS algorithm was executed 51 times, as dictated by the competition rules. Each run was continued for $T = 10000 \times D$ fitness evaluations. All the experiments were coded in Java, although we used the C/C++ original implementation of the CEC 2013 benchmark which was wrapped through the Java Native Interface (JNI) [24]. The 28×51 experiments were run, in parallel, on a heterogeneous network of Linux/Mac computers, through the distributed platform Kimeme [25].

A. Detailed results of CMA-ES-RIS

Tables III, IV and V show, respectively, the numerical results obtained with $D = 10$, $D = 30$ and $D = 50$ dimensions. For each test function, we report the best, worst, median and mean fitness value obtained at the end of the computational budget T , together with its standard deviation over the 51 runs available.

It can be seen that, apart from **f3** and **f5** (only in 50D), which however show a small deviation from the global optimum, all the unimodal functions (**f1** – **f5**) are solved in all the three dimensionalities considered. As for the multimodal functions (**f6** – **f20**), in 10 dimensions the optimum is detected in 4 cases, while the mean error ranges from $1.24e - 02$ to $6.17e + 02$ and the standard deviation ranges from $7.49e - 02$ to $1.72e + 02$, depending on the benchmark function. In 30 dimensions, the optimum is detected in 2 cases, while the mean error ranges from $6.94e - 04$ to $3.13e + 03$ and the standard deviation ranges from $5.41e - 03$ to $4.53e + 02$. In 50 dimensions, the optimum is detected in 1 case, while the mean error ranges from $8.60e - 03$ to $6.30e + 03$ and the standard deviation ranges from $5.86e - 03$ to $6.74e + 02$. Finally, regarding the composition functions (**f21** – **f28**), the mean error is in the order of $1e + 02$ to $1e + 03$ for all the three dimensionalities, while the standard deviation ranges from $2.27e + 01$ to $1.89e + 02$, from $8.41e + 00$ to $1.32e + 03$, and from $3.25e - 02$ to $1.53e + 03$, respectively for 10, 30 and 50 dimensions.

It can be seen that, in case of the unimodal function **f1**, CMA-ES-RIS is able to quickly converge to the optimum for all the problem dimensionalities considered. As for the multimodal function **f16**, CMA-ES-RIS obtains, fast and reliably, a rather small average error. On the other hand, on the composition function **f24** the error is relatively large.

B. Comparison against state-of-the-art algorithms

To provide a deeper understanding of the performance of CMA-ES-RIS, we compared the results shown in the previous

TABLE I. HOLM-BONFERRONI TEST, REFERENCE = CMA-ES-RIS
(RANK = 3.13E+00)

j	Optimizer	Rank	z_j	p_j	δ/j	Hypothesis
1	MDE-pBX	2.64e+00	-3.16e+00	7.80e-04	5.00e-02	Rejected
2	CCPSO2	2.24e+00	-5.79e+00	3.60e-09	2.50e-02	Rejected
3	CMA-ES	1.79e+00	-8.72e+00	1.41e-18	1.67e-02	Rejected

section - on the same CEC 2013 benchmark and under the same conditions of budget and repetitions - with three different state-of-the-art optimization algorithms, namely the original CMA-ES [13], the Self-Adaptive Differential Evolution with pBX crossover (MDE-pBX) [26], and the Cooperatively Coevolving Particle Swarms Optimizer (CCPSO2) [27]. As suggested in their original papers, MDE-pBX was run with population size equal to 100 individuals and group size q equal to 15% of the population size, while CCPSO2 was executed with population size equal to 30 individuals, Cauchy/Gaussian sampling selection probability $p = 0.5$ and set of potential group sizes $S = \{2, 5\}$, $S = \{2, 5, 10\}$, $S = \{2, 5, 10, 25\}$, for experiments in 10, 30 and 50 dimensions, respectively. To provide a fair comparison, CMA-ES and CMA-ES-RIS were configured with the same values of λ , μ , σ (see next section).

Tables VI, VII and VIII, show the average fitness error (with respect to the target fitness) and its standard deviation, over 51 repetitions, obtained by the four algorithms under comparison after $T = 10000 \times D$ fitness evaluations. We report also the result of the Wilcoxon test [28] applied, with confidence level 0.95, to each pair-wise comparison on the final values obtained by CMA-ES-RIS against CMA-ES, MDE-pBX and CCPSO2 (“+”, “-” and “=” indicate, respectively, a better, worse or equivalent performance of CMA-ES-RIS against the other algorithms). It can be seen that CMA-ES-RIS is extremely competitive: indeed, it outperforms MDE-pBX in 12 cases in 10D, 16 cases in 30D, and 18 cases in 50D, thus showing an excellent scalability. Most impressively, CMA-ES-RIS systematically outperforms (with few exceptions) both CMA-ES and CCPSO2 in all the three dimensionalities. This result is confirmed by the Holm-Bonferroni procedure [29], which we performed as described in [30], with confidence level 0.05. As shown in Table I, CMA-ES-RIS ranks first among the four algorithms, with the null-hypothesis (that CMA-ES-RIS and one of the other algorithms have indistinguishable performances) rejected in all the pair-wise comparisons.

C. Parameters

The CMA-ES-RIS algorithm was executed with the following parameters. As for the CMA-ES part of the algorithm, we used the default parameter setting of the original Java implementation [31]. In particular, the evolution strategy was configured with the standard values $\lambda = \lfloor 4 + 3 \ln(D) \rfloor$, $\mu = \lfloor \lambda/2 \rfloor$, and initial step-size $\sigma = 0.2$, which are broadly accepted in the literature.

The RIS algorithm has instead three main parameters which affect its performance, namely the inheritance factor α_e , the initial search radius ρ and the stop-threshold ε . As said, the inheritance factor represents the approximate percentage of genes transferred during re-sampling, with probability 0.5, from the elite to the trial solution. Its value ranges in $(0, 1]$: the smaller α_e , the lower is the number of genes copied from the best individual, and vice versa. If one needs to improve

the exploration capabilities of the algorithm, a smaller value of α_e should be used: this might be needed, for example, to tackle highly multi-modal landscapes. Otherwise, a larger value would increase the exploitation pressure.

Similarly, possible values for the fraction δ of the search space (used to initialize the search radius ρ in the exploitative local search) are $(0, 0.5]$, so that smaller values make the local search explore in a narrow neighbourhood of the current solution $\mathbf{x}_s$, and vice versa. This parameter should be related to the width of the basin(s) of attraction present in the fitness landscape, so that, if one has some prior knowledge about the fitness, the initial search radius should be as large as the smaller basin.

As for the stop-threshold ε , its value obviously affects the exploitation pressure of the local search: smaller values let RIS refine the search deeper, larger ones instead produce a coarser exploitation stage in favor of exploration. This parameter depends on the precision one wants to obtain on the fitness, and is related to the steepness (or, on the contrary, flatness) of the fitness landscape. Therefore ε may take arbitrarily small values, depending on the required precision.

Finally, the most crucial – and sensitive – parameter in the CMA-ES-RIS structure is the budget ratio between r_{CMA-ES} which determines the amount of budget assigned to both CMA-ES and RIS. This parameter ranges, theoretically, in $(0, 1)$. However, according to our empirical observation in order to have a well-balanced combination of the two algorithms this parameter should be selected within the range $[0.2, 0.5]$. A value larger than 0.5 would obviously assign more budget to the super-fit individual generation. As shown in the comparative experiments, the RIS application has an important impact on the performance with respect to the sole CMA-ES action, see Section III-B. This parameter should be set based both on theoretical and practical considerations: on one hand, allotting more budget to CMA-ES means that more effort is put in the first stage to learn the variable linkages and determine a super-fit individual; on the other, due to the higher complexity of CMA-ES with respect to RIS (see below), a larger value of r_{CMA-ES} increases the overall complexity of CMA-ES-RIS.

The numerical results shown in the previous section were obtained with $\alpha_e = 0.5$, $\delta = 0.2$, $\varepsilon = 10^{-6}$, and $r_{CMA-ES} = 0.3$. These values were selected tuning each parameter keeping the others at a fixed value chosen from a predefined set. More specifically, we tested the following sets of values: $\alpha_e = \{0.2, 0.5, 0.8\}$, $\delta = \{0.2, 0.3, 0.5\}$, $\varepsilon = \{10^{-10}, 10^{-6}, 10^{-4}\}$ and $r_{CMA-ES} = \{0.2, 0.3, 0.5\}$. We recorded the average fitness error (over 51 runs) obtained with CMA-ES-RIS on the whole CEC 2013 benchmark at the three different dimensionalities, and then selected, for each parameter, the value showing the best global mean fitness error. The performance obtained with the best parameter setting can be explained as follows: $\alpha_e = 0.5$ forces the trial solution to inherit at least half genome from the elite (with probability 0.5), thus guaranteeing a well-balanced inheritance ratio. Moreover, configuring the local search so that it explores a neighbourhood equal to $1/5$ of the whole search space provides a sufficiently large space for fitness improvements, still being enough focused on a restricted area of the search space. Similarly, the value $\varepsilon = 10^{-6}$ guarantees a good trade-off between the re-sampling with inheritance and the exploitation local search, as well as

$r_{CMA-ES} = 0.3$ guarantees the best trade-off between CMA-ES and RIS, allowing an optimal usage of the budget.

D. Algorithm complexity

Table II reports four measures of the computational complexity of the proposed CMA-ES-RIS, as defined in [23]. More specifically, T_0 is the execution time of a benchmark program performing some simple mathematical operations in double precision, T_1 is the time needed to compute 200000 evaluations of the benchmark function **f14**, and $\widehat{T_2}$ it the average execution time (over 5 repetitions) of a run of CMA-ES-RIS performing 200000 evaluations of the benchmark function **f3**. T_1 , T_1 and $\widehat{T_2}$ are here expressed in ms. A fourth, adimensional metric, is also defined as $(\widehat{T_2} - T_1)/T_0$. To show how the algorithm complexity changes with the problem dimension, T_1 , $\widehat{T_2}$, $(\widehat{T_2} - T_1)/T_0$ are reported for the three dimensionalities considered in this study (10, 30 and 50).

TABLE II. COMPUTATIONAL COMPLEXITY

	T_0	T_1	$\widehat{T_2}$	$(\widehat{T_2} - T_1)/T_0$
$D = 10$	27.0	544.0	496.0	-1.8
$D = 30$		914.0	2451.6	56.9
$D = 50$		1520.0	6390.2	180.4

As in our previous experiments, the algorithm, as well as the program to compute its complexity, was coded in Java, wrapping the C/C++ implementation of the benchmark functions through JNI. The program was executed, in single thread, on a Mac OS X system, version 10.7.5, with 2.2 GHz Intel Core i7 processor, and 4 GB RAM. The timestamps were measured through the Java system call `System.currentTimeMillis()`.

It can be seen that the time complexity of CMA-ES-RIS grows super-linearly (but sub-quadratically) with the problem dimensionality, showing a trade-off between the complexity of CMA-ES, which is quadratic, and that of RIS, which is linear, as proven previously. For the sake of completeness, it is interesting to note the also the space complexity is quadratic for the CMA-ES part (because of the covariance matrix, which requires n^2 elements), and linear for RIS (which instead requires only 3 n -dimensional vectors, i.e. $\mathbf{x}_e$, $\mathbf{x}_s$ and $\mathbf{x}_t$). Thus the proposed approach is mostly indicated for low-medium scale problems, while for large-scale problems the first part of the algorithm, based on CMA-ES, becomes naturally more time consuming.

IV. CONCLUSION

In this paper we introduced CMA-ES-RIS, a simple yet powerful optimization algorithm composed of two sequential elements: an initial CMA-ES step which is used to generate a super-fit individual, and a local search with inheritance which improves upon it. In the first stage, thanks to the covariance matrix adaptation mechanism, the features of the fitness landscape are implicitly analysed and exploited to rapidly converge to a promising area of the search space. Once this area is detected, the best individual in the final population of CMA-ES is selected as a starting point for the local search algorithm, which refines the search further. During this second stage, based on a fitness threshold, the algorithm is however

allowed to perform a re-sampling with partial inheritance of the current best individual, so to prevent premature convergence and explore other areas of the search space. We presented the numerical results obtained on the CEC 2013 real-parameter optimization benchmark and discussed the parameter setting and time complexity of the proposed approach, showing that CMA-ES-RIS is able to obtain good solutions fast and reliably on a broad range of optimization problems. In the future, the CMA-ES-RIS structure will be further investigated testing different super-fit mechanisms and local search methods.

TABLE III. RESULTS FOR 10D

Func.	Best	Worst	Median	Mean	Std
1	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
3	0.00e+00	3.29e+01	0.00e+00	7.04e-01	4.57e+00
4	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
5	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
6	0.00e+00	9.81e+00	2.42e-03	1.10e+00	2.83e+00
7	1.99e+00	1.78e+02	4.01e+01	5.33e+01	4.63e+01
8	2.00e+01	2.06e+01	2.03e+01	2.03e+01	1.36e-01
9	1.10e+00	5.48e+00	3.63e+00	3.59e+00	1.03e+00
10	0.00e+00	5.17e-02	9.86e-03	1.24e-02	1.33e-02
11	0.00e+00	7.96e+00	2.98e+00	3.57e+00	1.46e+00
12	1.99e+00	2.59e+01	1.29e+01	1.29e+01	5.36e+00
13	4.48e+00	5.00e+01	2.64e+01	2.56e+01	1.07e+01
14	3.60e+00	2.60e+02	1.13e+02	1.02e+02	7.32e+01
15	1.42e+02	1.03e+03	6.33e+02	6.17e+02	1.72e+02
16	0.00e+00	3.55e-01	1.76e-01	1.64e-01	7.49e-02
17	1.69e+00	1.79e+01	1.18e+01	1.04e+01	3.70e+00
18	1.95e+01	4.59e+01	2.95e+01	2.98e+01	6.10e+00
19	3.94e-01	1.62e+00	7.61e-01	8.14e-01	2.71e-01
20	3.35e+00	4.70e+00	4.25e+00	4.16e+00	3.95e-01
21	1.00e+02	4.00e+02	2.00e+02	1.61e+02	5.97e+01
22	5.05e+01	4.77e+02	2.68e+02	2.44e+02	1.07e+02
23	3.40e+02	1.23e+03	8.34e+02	8.35e+02	1.89e+02
24	1.07e+02	1.32e+02	1.18e+02	1.19e+02	5.63e+00
25	1.14e+02	2.17e+02	2.07e+02	1.93e+02	3.39e+01
26	1.04e+02	2.00e+02	2.00e+02	1.61e+02	4.02e+01
27	2.25e+02	4.00e+02	3.08e+02	3.13e+02	2.27e+01
28	1.74e-06	3.00e+02	3.00e+02	2.06e+02	1.06e+02

TABLE IV. RESULTS FOR 30D

Func.	Best	Worst	Median	Mean	Std
1	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
3	0.00e+00	7.73e+04	9.65e-01	2.24e+03	1.09e+04
4	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
5	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
6	0.00e+00	9.78e-03	0.00e+00	6.94e-04	1.99e-03
7	1.03e+00	1.25e+02	3.79e+01	4.48e+01	2.93e+01
8	2.07e+01	2.10e+01	2.09e+01	2.09e+01	8.11e-02
9	1.99e+01	2.76e+01	2.38e+01	2.37e+01	1.93e+00
10	0.00e+00	2.46e-02	7.40e-03	8.31e-03	5.41e-03
11	1.49e+01	4.97e+01	2.39e+01	2.54e+01	6.30e+00
12	2.79e+01	1.61e+02	5.77e+01	7.94e+01	4.35e+01
13	5.57e+01	2.41e+02	1.45e+02	1.56e+02	5.36e+01
14	3.27e+02	1.30e+03	8.08e+02	7.92e+02	2.19e+02
15	1.90e+03	4.60e+03	3.12e+03	3.13e+03	4.53e+02
16	2.28e-02	3.17e-01	9.03e-02	1.07e-01	6.71e-02
17	4.00e+01	6.71e+01	5.49e+01	5.50e+01	5.19e+00
18	1.34e+02	2.44e+02	1.89e+02	1.89e+02	2.71e+01
19	1.54e+00	4.21e+00	2.71e+00	2.80e+00	6.35e-01
20	1.23e+01	1.45e+01	1.45e+01	1.43e+01	5.69e-01
21	1.00e+02	3.00e+02	2.00e+02	1.86e+02	3.97e+01
22	6.34e+02	1.74e+03	1.18e+03	1.17e+03	2.90e+02
23	2.71e+03	5.35e+03	4.01e+03	4.03e+03	5.38e+02
24	1.97e+02	2.80e+02	2.65e+02	2.59e+02	1.74e+01
25	2.59e+02	3.01e+02	2.82e+02	2.82e+02	8.41e+00
26	1.32e+02	2.00e+02	2.00e+02	1.97e+02	1.20e+01
27	4.00e+02	1.01e+03	8.15e+02	7.49e+02	1.85e+02
28	1.00e+02	9.57e+03	3.00e+02	5.39e+02	1.32e+03

ACKNOWLEDGMENT

INCAS³ is co-funded by the Province of Drenthe, the Municipality of Assen, the European Fund for Regional Development and the Ministry of Economic Affairs, Peaks in the Delta. This research is supported by the Academy of Finland, Akatemiututkija 130600, “Algorithmic design issues in Memetic Computing” and Tutkijatohtori 140487, “Algorithmic Design and Software Implementation: a Novel Optimization Platform”. The numerical experiments have been

TABLE V. RESULTS FOR 50D

Func.	Best	Worst	Median	Mean	Std
1	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
2	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
3	9.56e-05	3.96e+06	8.55e+03	2.83e+05	7.80e+05
4	0.00e+00	0.00e+00	0.00e+00	0.00e+00	0.00e+00
5	1.29e-08	7.64e-08	3.26e-08	3.54e-08	1.44e-08
6	0.00e+00	4.34e+01	3.52e+00	9.51e+00	1.41e+01
7	1.39e+01	1.09e+02	4.50e+01	4.81e+01	2.16e+01
8	2.09e+01	2.12e+01	2.11e+01	2.10e+01	6.02e-02
9	4.21e+01	5.37e+01	4.69e+01	4.72e+01	2.76e+00
10	0.00e+00	2.71e-02	9.86e-03	8.60e-03	5.86e-03
11	2.89e+01	9.55e+01	5.07e+01	5.36e+01	1.31e+01
12	7.46e+01	4.28e+02	2.91e+02	2.66e+02	1.02e+02
13	2.04e+02	5.94e+02	4.70e+02	4.56e+02	8.54e+01
14	8.13e+02	2.40e+03	1.49e+03	1.49e+03	3.07e+02
15	4.60e+03	8.73e+03	6.38e+03	6.30e+03	6.74e+02
16	2.37e-02	2.02e-01	7.76e-02	8.66e-02	4.20e-02
17	8.39e+01	1.24e+02	1.01e+02	1.02e+02	1.04e+01
18	2.89e+02	5.43e+02	4.18e+02	4.18e+02	5.24e+01
19	2.97e+00	6.86e+00	4.88e+00	5.04e+00	9.33e-01
20	2.11e+01	2.45e+01	2.45e+01	2.43e+01	5.49e-01
21	1.00e+02	8.36e+02	2.00e+02	2.85e+02	2.20e+02
22	1.48e+03	3.34e+03	2.36e+03	2.39e+03	3.67e+02
23	6.73e+03	1.19e+04	8.27e+03	8.37e+03	1.01e+03
24	2.70e+02	3.50e+02	3.29e+02	3.22e+02	1.95e+01
25	3.45e+02	3.94e+02	3.67e+02	3.66e+02	9.94e+00
26	2.00e+02	2.00e+02	2.00e+02	2.00e+02	3.25e-02
27	4.00e+02	1.58e+03	1.24e+03	1.25e+03	2.08e+02
28	4.00e+02	7.25e+03	4.00e+02	1.24e+03	1.53e+03

carried out on the computer network of the De Montfort University by means of the software for distributed optimization Kimeme [25].

REFERENCES

- [1] D. H. Wolpert and W. G. Macready, "No free lunch theorems for optimization," *IEEE Transactions on Evolutionary Computation*, vol. 1, no. 1, pp. 67–82, 1997.
- [2] D. Molina, M. Lozano, C. Garcia-Martinez, and F. Herrera, "Memetic Algorithms for Continuous Optimization Based on Local Search Chains," *Evolutionary Computation*, vol. 18, no. 1, pp. 27–63, 2010.
- [3] M. A. Montes de Oca, T. Stutzle, M. Birattari, and M. Dorigo, "Frankenstein's PSO: A Composite Particle Swarm Optimization Algorithm," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 5, pp. 1120–1132, 2009.
- [4] A. K. Qin, V. L. Huang, and P. N. Suganthan, "Differential Evolution Algorithm With Strategy Adaptation for Global Numerical Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 398–417, 2009.
- [5] J. A. Vrugt, B. A. Robinson, and J. M. Hyman, "Self-Adaptive Multithreshold Search for Global Optimization in Real-Parameter Spaces," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 2, pp. 243–259, 2009.
- [6] F. Peng, K. Tang, G. Chen, and X. Yao, "Population-Based Algorithm Portfolios for Numerical Optimization," *IEEE Transactions on Evolutionary Computation*, vol. 14, no. 5, pp. 782–800, 2010.
- [7] G. Iacca, F. Neri, E. Mininno, Y. S. Ong, and M. H. Lim, "Ockham's Razor in Memetic Computing: Three Stage Optimal Memetic Exploration," *Information Sciences*, vol. 188, pp. 17–43, 2012.
- [8] F. Caraffini, F. Neri, G. Iacca, and A. Mol, "Parallel memetic structures," *Information Sciences*, vol. 227, pp. 60–82, 2013.
- [9] A. Caponio, F. Neri, and V. Tirronen, "Super-fit control adaptation in memetic differential evolution frameworks," *Soft Computing - A Fusion of Foundations, Methodologies and Applications*, vol. 13, pp. 811–831, 2009.
- [10] G. Iacca, R. Mallipeddi, E. Mininno, F. Neri, and P. N. Suganthan, "Super-fit and Population Size Reduction Mechanisms in Compact Differential Evolution," in *Proceedings of IEEE Symposium on Memetic Computing*, 2011, pp. 21–28.
- [11] N. Hansen and A. Ostermeier, "Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation," in *Proceedings of the IEEE International Conference on Evolutionary Computation*, 1996, pp. 312–317.
- [12] —, "Completely derandomized self-adaptation in evolution strategies," *Evolutionary Computation*, vol. 9, no. 2, pp. 159–195, 2001.
- [13] N. Hansen, S. D. Müller, and P. Koumoutsakos, "Reducing the Time Complexity of the Derandomized Evolution Strategy with Covariance Matrix Adaptation (CMA-ES)," *Evolutionary Computation*, vol. 11, no. 1, pp. 1–18, 2003.
- [14] H.-P. Schwefel, "Kybernetische Evolution als Strategie der experimentellen Forschung in der Strömungstechnik," Ph.D. dissertation, Technical University of Berlin, Hermann Föttinger-Institute for Fluid Dynamics, 1965.
- [15] I. Rechenberg, "Evolutionstrategie – Optimierung technischer Systeme nach Prinzipien der biologischen Evolution," Ph.D. dissertation, Technical University of Berlin, 1971.
- [16] A. Auger and N. Hansen, "A Restart CMA Evolution Strategy With Increasing Population Size," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2005, pp. 1769–1776.
- [17] S. Ghosh, S. Roy, S. Islam, S. Das, and P. Suganthan, "A differential covariance matrix adaptation evolutionary algorithm for global optimization," in *Differential Evolution (SDE), 2011 IEEE Symposium on*, april 2011, pp. 1–8.
- [18] C. Igel, T. Suttorp, and N. Hansen, "A Computational Efficient Covariance Matrix Update and a (1+1)-CMA for Evolution Strategies," in *Proceedings of the Genetic and Evolutionary Computation Conference*. ACM Press, 2006, pp. 453–460.
- [19] W. Dong and X. Yao, "Covariance matrix repairing in Gaussian based EDAs," in *IEEE Congress on Evolutionary Computation*, 2007, pp. 415–422.
- [20] F. Neri and V. Tirronen, "Recent Advances in Differential Evolution: A Review and Experimental Analysis," *Artificial Intelligence Review*, vol. 33, no. 1–2, pp. 61–106, 2010.
- [21] F. Neri, G. Iacca, and E. Mininno, "Disturbed Exploitation compact Differential Evolution for Limited Memory Optimization Problems," *Information Sciences*, vol. 181, no. 12, pp. 2469–2487, 2011.
- [22] L.-Y. Tseng and C. Chen, "Multiple trajectory search for Large Scale Global Optimization," in *Proceedings of the IEEE Congress on Evolutionary Computation*, 2008, pp. 3052–3059.
- [23] J. J. Liang, B. Y. Qu, P. N. Suganthan, and A. G. Hernandez-Daz, "Problem Definitions and Evaluation Criteria for the CEC 2013 Special Session on Real-Parameter Optimization," Zhengzhou University and Nanyang Technological University, Zhengzhou China and Singapore, Tech. Rep. 201212, 2013.
- [24] L. Sheng, *The Java Native Interface: Programmer's Guide and Specification (The Java Series)*. Addison-Wesley, 1999.
- [25] Cyber Dyne Srl Home Page, "Kimeme," 2012, <http://cyberdynesoft.it/>.
- [26] S. Islam, S. Das, S. Ghosh, S. Roy, and P. Suganthan, "An Adaptive Differential Evolution Algorithm With Novel Mutation and Crossover Strategies for Global Numerical Optimization," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 42, no. 2, pp. 482–500, april 2012.
- [27] X. Li and X. Yao, "Cooperatively Coevolving Particle Swarms for Large Scale Optimization," *Evolutionary Computation, IEEE Transactions on*, vol. 16, no. 2, pp. 210–224, april 2012.
- [28] F. Wilcoxon, "Individual comparisons by ranking methods," *Biometrics Bulletin*, vol. 1, no. 6, pp. 80–83, 1945.
- [29] S. Holm, "A simple sequentially rejective multiple test procedure," *Scandinavian Journal of Statistics*, vol. 6, no. 2, pp. 65–70, 1979.
- [30] S. Garcia, A. Fernandez, J. Luengo, and F. Herrera, "A study of statistical techniques and performance measures for genetics-based machine learning: accuracy and interpretability," *Soft Computing*, vol. 13, no. 10, pp. 959–977, 2008.
- [31] N. Hansen, "The CMA Evolution Strategy," 2012, <http://www.lri.fr/~hansen/cmaesintro.html>.

TABLE VI. AVERAGE ERROR $\pm$ STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REFERENCE=CMA-ES-RIS) FOR CMA-ES-RIS AGAINST STATE-OF-THE-ART ALGORITHMS IN 10D

	CMA-ES-RIS	MDE-pBX		CMA-ES		CCPSO2	
f_1	0.00e+00 $\pm$ 0.00e+00	0.00e+00 $\pm$ 0.00e+00	=	0.00e+00 $\pm$ 0.00e+00	=	1.92e-04 $\pm$ 1.16e-03	+
f_2	0.00e+00 $\pm$ 0.00e+00	4.15e+02 $\pm$ 9.60e+02	+	0.00e+00 $\pm$ 0.00e+00	=	9.93e+05 $\pm$ 7.58e+05	+
f_3	7.04e-01 $\pm$ 4.57e+00	4.96e+03 $\pm$ 4.66e+04	+	5.69e-01 $\pm$ 1.81e+00	=	2.13e+07 $\pm$ 3.13e+07	+
f_4	0.00e+00 $\pm$ 0.00e+00	6.50e-02 $\pm$ 6.38e-01	+	0.00e+00 $\pm$ 0.00e+00	=	8.80e+03 $\pm$ 2.50e+03	+
f_5	0.00e+00 $\pm$ 2.76e-14	0.00e+00 $\pm$ 7.54e-14	+	0.00e+00 $\pm$ 0.00e+00	=	2.94e-03 $\pm$ 8.06e-03	+
f_6	1.10e+00 $\pm$ 2.85e+00	6.18e+00 $\pm$ 4.73e+00	+	6.74e+00 $\pm$ 6.74e+00	+	1.52e+00 $\pm$ 2.91e+00	+
f_7	5.33e+01 $\pm$ 4.63e+01	5.63e+00 $\pm$ 7.94e+00	-	5.45e+08 $\pm$ 5.38e+09	=	3.27e+01 $\pm$ 8.31e+00	=
f_8	2.03e+01 $\pm$ 1.36e-01	2.05e+01 $\pm$ 1.06e-01	+	2.03e+01 $\pm$ 1.32e-01	=	2.04e+01 $\pm$ 7.66e-02	+
f_9	3.59e+00 $\pm$ 1.03e+00	2.37e+00 $\pm$ 1.41e+00	-	1.50e+01 $\pm$ 3.65e+00	+	4.98e+00 $\pm$ 9.13e-01	+
f_{10}	1.24e-02 $\pm$ 1.33e-02	1.25e-01 $\pm$ 9.20e-02	+	1.33e-02 $\pm$ 1.39e-02	=	1.47e+00 $\pm$ 6.44e-01	+
f_{11}	3.57e+00 $\pm$ 1.46e+00	2.48e+00 $\pm$ 1.61e+00	-	2.31e+02 $\pm$ 2.67e+02	+	1.97e+00 $\pm$ 1.18e+00	-
f_{12}	1.29e+01 $\pm$ 5.36e+00	1.06e+01 $\pm$ 4.76e+00	-	3.49e+02 $\pm$ 3.81e+02	+	2.64e+01 $\pm$ 7.93e+00	+
f_{13}	2.56e+01 $\pm$ 1.07e+01	2.01e+01 $\pm$ 7.94e+00	-	2.94e+02 $\pm$ 3.99e+02	+	3.56e+01 $\pm$ 8.30e+00	+
f_{14}	1.02e+02 $\pm$ 7.32e+01	1.25e+02 $\pm$ 1.12e+02	=	1.88e+03 $\pm$ 4.25e+02	+	5.27e+01 $\pm$ 3.86e+01	-
f_{15}	6.17e+02 $\pm$ 1.72e+02	7.26e+02 $\pm$ 2.61e+02	+	1.80e+03 $\pm$ 3.92e+02	+	8.92e+02 $\pm$ 2.18e+02	+
f_{16}	1.64e-01 $\pm$ 7.49e-02	5.43e-01 $\pm$ 4.49e-01	+	4.51e-01 $\pm$ 5.06e-01	+	1.18e+00 $\pm$ 2.19e-01	+
f_{17}	1.04e+01 $\pm$ 3.70e+00	1.32e+01 $\pm$ 1.85e+00	+	9.55e+02 $\pm$ 3.42e+02	+	1.60e+01 $\pm$ 2.07e+00	+
f_{18}	2.98e+01 $\pm$ 6.10e+00	1.93e+01 $\pm$ 4.67e+00	-	9.01e+02 $\pm$ 3.10e+02	+	5.41e+01 $\pm$ 6.33e+00	+
f_{19}	8.14e-01 $\pm$ 2.71e-01	6.44e-01 $\pm$ 2.09e-01	-	1.19e+00 $\pm$ 5.00e-01	+	7.65e-01 $\pm$ 2.52e-01	=
f_{20}	4.16e+00 $\pm$ 3.95e-01	2.87e+00 $\pm$ 5.25e-01	-	4.68e+00 $\pm$ 3.79e-01	+	3.50e+00 $\pm$ 1.96e-01	-
f_{21}	1.61e+02 $\pm$ 5.97e+01	3.98e+02 $\pm$ 1.99e+01	+	3.68e+02 $\pm$ 8.71e+01	+	3.73e+02 $\pm$ 6.75e+01	+
f_{22}	2.44e+02 $\pm$ 1.07e+02	1.33e+02 $\pm$ 1.04e+02	-	2.32e+03 $\pm$ 4.25e+02	+	7.27e+01 $\pm$ 4.98e+01	-
f_{23}	8.35e+02 $\pm$ 1.89e+02	8.82e+02 $\pm$ 3.08e+02	=	2.25e+03 $\pm$ 4.48e+02	+	1.15e+03 $\pm$ 2.58e+02	+
f_{24}	1.19e+02 $\pm$ 5.63e+00	2.02e+02 $\pm$ 1.56e+01	+	3.83e+02 $\pm$ 1.57e+02	+	2.01e+02 $\pm$ 2.54e+01	+
f_{25}	1.93e+02 $\pm$ 3.39e+01	2.00e+02 $\pm$ 1.23e+01	+	2.62e+02 $\pm$ 4.92e+01	+	2.12e+02 $\pm$ 1.09e+01	+
f_{26}	1.61e+02 $\pm$ 4.02e+01	1.47e+02 $\pm$ 4.36e+01	-	2.57e+02 $\pm$ 1.13e+02	+	1.58e+02 $\pm$ 2.40e+01	=
f_{27}	3.13e+02 $\pm$ 2.27e+01	3.06e+02 $\pm$ 2.76e+01	-	4.23e+02 $\pm$ 1.35e+02	+	4.27e+02 $\pm$ 4.91e+01	+
f_{28}	2.06e+02 $\pm$ 1.06e+02	3.07e+02 $\pm$ 5.78e+01	=	1.21e+03 $\pm$ 1.21e+03	+	3.01e+02 $\pm$ 1.28e+02	+

TABLE VII. AVERAGE ERROR $\pm$ STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REFERENCE=CMA-ES-RIS) FOR CMA-ES-RIS AGAINST STATE-OF-THE-ART ALGORITHMS IN 30D

	CMA-ES-RIS	MDE-pBX		CMA-ES		CCPSO2	
f_1	0.00e+00 $\pm$ 1.68e-13	0.00e+00 $\pm$ 4.09e-13	=	0.00e+00 $\pm$ 5.57e-14	=	2.27e-13 $\pm$ 1.21e-12	+
f_2	0.00e+00 $\pm$ 1.83e-13	9.56e+04 $\pm$ 6.16e+04	+	0.00e+00 $\pm$ 7.88e-14	=	9.95e+05 $\pm$ 5.24e+05	+
f_3	2.24e+03 $\pm$ 1.09e+04	1.80e+07 $\pm$ 3.12e+07	+	1.53e+01 $\pm$ 1.23e+02	-	5.59e+08 $\pm$ 5.57e+08	+
f_4	0.00e+00 $\pm$ 1.68e-13	1.32e+01 $\pm$ 5.46e+01	+	0.00e+00 $\pm$ 5.08e-14	=	5.57e+04 $\pm$ 2.06e+04	+
f_5	2.88e-10 $\pm$ 4.04e-10	1.14e-13 $\pm$ 8.35e-13	+	1.14e-13 $\pm$ 1.14e-14	-	2.62e-08 $\pm$ 6.05e-08	+
f_6	6.94e-04 $\pm$ 1.99e-03	1.99e+01 $\pm$ 2.22e+01	+	3.05e+00 $\pm$ 8.33e+00	+	2.19e+01 $\pm$ 2.27e+01	+
f_7	4.48e+01 $\pm$ 2.93e+01	5.70e+01 $\pm$ 1.77e+01	+	9.83e+03 $\pm$ 6.44e+04	=	1.15e+02 $\pm$ 3.11e+01	+
f_8	2.09e+01 $\pm$ 8.11e-02	2.11e+01 $\pm$ 5.94e-02	+	2.09e+01 $\pm$ 6.66e-02	+	2.10e+01 $\pm$ 4.60e-02	+
f_9	2.37e+01 $\pm$ 1.93e+00	2.22e+01 $\pm$ 4.80e+00	-	4.45e+01 $\pm$ 6.99e+00	+	2.84e+01 $\pm$ 2.08e+00	+
f_{10}	8.31e-03 $\pm$ 5.41e-03	1.64e-01 $\pm$ 1.20e-01	+	1.73e-02 $\pm$ 1.25e-02	+	1.48e-01 $\pm$ 6.90e-02	+
f_{11}	2.54e+01 $\pm$ 6.30e+00	4.62e+01 $\pm$ 1.44e+01	+	9.47e+01 $\pm$ 2.36e+02	+	1.19e-01 $\pm$ 2.90e-01	-
f_{12}	7.94e+01 $\pm$ 4.35e+01	6.94e+01 $\pm$ 2.00e+01	+	7.11e+02 $\pm$ 9.66e+02	=	2.12e+02 $\pm$ 5.24e+01	+
f_{13}	1.56e+02 $\pm$ 5.36e+01	1.49e+02 $\pm$ 3.66e+01	+	1.64e+03 $\pm$ 1.66e+03	+	2.44e+02 $\pm$ 3.44e+01	+
f_{14}	7.92e+02 $\pm$ 2.19e+02	1.17e+03 $\pm$ 3.95e+02	+	5.21e+03 $\pm$ 7.37e+02	+	4.48e+00 $\pm$ 2.87e+00	-
f_{15}	3.13e+03 $\pm$ 4.53e+02	3.95e+03 $\pm$ 6.57e+02	+	5.37e+03 $\pm$ 6.73e+02	+	3.85e+03 $\pm$ 4.52e+02	+
f_{16}	1.07e-01 $\pm$ 6.71e-02	1.25e+00 $\pm$ 6.19e-01	+	1.26e-01 $\pm$ 8.87e-02	=	2.16e+00 $\pm$ 3.76e-01	+
f_{17}	5.50e+01 $\pm$ 5.19e+00	7.05e+01 $\pm$ 1.24e+01	+	3.95e+03 $\pm$ 7.89e+02	+	3.07e+01 $\pm$ 3.03e+00	-
f_{18}	1.89e+02 $\pm$ 2.71e+01	8.26e+01 $\pm$ 1.89e+01	-	4.23e+03 $\pm$ 8.31e+02	+	2.31e+02 $\pm$ 5.43e+01	+
f_{19}	2.80e+00 $\pm$ 6.35e-01	9.54e+00 $\pm$ 5.54e+00	+	3.66e+00 $\pm$ 9.52e-01	+	7.77e-01 $\pm$ 1.58e-01	-
f_{20}	1.43e+01 $\pm$ 5.69e-01	1.07e+01 $\pm$ 7.75e-01	-	1.50e+01 $\pm$ 6.45e-02	+	1.35e+01 $\pm$ 5.50e-01	-
f_{21}	1.86e+02 $\pm$ 3.97e+01	3.40e+02 $\pm$ 7.62e+01	+	3.05e+02 $\pm$ 9.01e+01	+	2.37e+02 $\pm$ 6.71e+01	+
f_{22}	1.17e+03 $\pm$ 2.90e+02	1.17e+03 $\pm$ 4.92e+02	=	6.97e+03 $\pm$ 1.06e+03	+	9.87e+01 $\pm$ 6.70e+01	-
f_{23}	4.03e+03 $\pm$ 5.38e+02	4.70e+03 $\pm$ 7.70e+02	+	6.76e+03 $\pm$ 6.82e+02	+	4.99e+03 $\pm$ 6.31e+02	+
f_{24}	2.59e+02 $\pm$ 1.74e+01	2.31e+02 $\pm$ 8.60e+00	-	8.19e+02 $\pm$ 6.15e+02	+	2.80e+02 $\pm$ 6.34e+00	+
f_{25}	2.82e+02 $\pm$ 8.41e+00	2.79e+02 $\pm$ 1.38e+01	-	3.46e+02 $\pm$ 1.45e+02	+	2.98e+02 $\pm$ 6.94e+00	+
f_{26}	1.97e+02 $\pm$ 1.20e+01	2.26e+02 $\pm$ 5.15e+01	+	5.51e+02 $\pm$ 5.14e+02	+	2.00e+02 $\pm$ 6.76e-01	+
f_{27}	7.49e+02 $\pm$ 1.85e+02	6.50e+02 $\pm$ 1.04e+02	-	8.53e+02 $\pm$ 2.42e+02	=	1.04e+03 $\pm$ 8.09e+01	+
f_{28}	5.39e+02 $\pm$ 1.32e+03	3.09e+02 $\pm$ 1.50e+02	-	1.96e+03 $\pm$ 3.40e+03	=	4.35e+02 $\pm$ 5.10e+02	-

TABLE VIII. AVERAGE ERROR $\pm$ STANDARD DEVIATION AND WILCOXON RANK-SUM TEST (REFERENCE=CMA-ES-RIS) FOR CMA-ES-RIS AGAINST STATE-OF-THE-ART ALGORITHMS IN 50D

	CMA-ES-RIS	MDE-pBX		CMA-ES		CCPSO2	
f_1	2.27e-13 $\pm$ 0.00e+00	1.11e-11 $\pm$ 7.68e-11	+	2.27e-13 $\pm$ 0.00e+00	=	6.82e-13 $\pm$ 1.31e-12	+
f_2	2.27e-13 $\pm$ 0.00e+00	4.37e+05 $\pm$ 1.64e+05	+	2.27e-13 $\pm$ 0.00e+00	=	1.85e+06 $\pm$ 9.33e+05	+
f_3	2.83e+05 $\pm$ 7.80e+05	8.45e+07 $\pm$ 1.46e+08	+	9.64e+02 $\pm$ 4.49e+03	-	1.98e+09 $\pm$ 2.09e+09	+
f_4	2.27e-13 $\pm$ 0.00e+00	3.05e+01 $\pm$ 6.62e+01	+	2.27e-13 $\pm$ 0.00e+00	=	1.00e+05 $\pm$ 3.57e+04	+
f_5	3.54e-08 $\pm$ 1.44e-08	1.93e-11 $\pm$ 9.76e-11	-	6.82e-13 $\pm$ 1.57e-12	-	2.70e-10 $\pm$ 8.03e-10	-
f_6	9.51e+00 $\pm$ 1.41e+01	5.48e+01 $\pm$ 2.12e+01	+	4.32e+01 $\pm$ 7.10e+00	+	4.35e+01 $\pm$ 1.36e+01	+
f_7	4.81e+01 $\pm$ 2.16e+01	6.59e+01 $\pm$ 1.06e+01	+	4.19e+01 $\pm$ 1.70e+01	=	1.37e+02 $\pm$ 2.31e+01	+
f_8	2.10e+01 $\pm$ 6.02e-02	2.12e+01 $\pm$ 4.44e-02	+	2.11e+01 $\pm$ 9.75e-02	+	2.11e+01 $\pm$ 4.49e-02	+
f_9	4.72e+01 $\pm$ 2.76e+00	4.32e+01 $\pm$ 7.71e+00	-	7.66e+01 $\pm$ 7.77e+00	+	5.79e+01 $\pm$ 4.39e+00	+
f_{10}	8.60e-03 $\pm$ 5.86e-03	1.34e-01 $\pm$ 1.23e-01	+	2.24e-02 $\pm$ 1.49e-02	+	1.24e-01 $\pm$ 4.62e-02	+
f_{11}	5.36e+01 $\pm$ 1.31e+01	1.24e+02 $\pm$ 2.87e+01	+	2.19e+02 $\pm$ 4.56e+02	+	4.31e-01 $\pm$ 5.74e-01	-
f_{12}	2.66e+02 $\pm$ 1.02e+02	1.58e+02 $\pm$ 3.25e+01	-	2.25e+03 $\pm$ 1.37e+03	+	4.46e+02 $\pm$ 7.92e+01	+
f_{13}	4.56e+02 $\pm$ 8.54e+01	3.24e+02 $\pm$ 4.74e+01	-	3.36e+03 $\pm$ 1.09e+03	+	5.49e+02 $\pm$ 6.67e+01	+
f_{14}	1.49e+03 $\pm$ 3.07e+02	2.65e+03 $\pm$ 8.86e+02	+	8.82e+03 $\pm$ 1.04e+03	+	6.45e+00 $\pm$ 3.20e+00	-
f_{15}	6.30e+03 $\pm$ 6.74e+02	7.46e+03 $\pm$ 7.95e+02	+	9.09e+03 $\pm$ 9.43e+02	+	7.95e+03 $\pm$ 7.11e+02	+
f_{16}	8.66e-02 $\pm$ 4.20e-02	1.75e+00 $\pm$ 7.40e-01	+	8.01e-02 $\pm$ 4.72e-02	=	2.39e+00 $\pm$ 5.90e-01	+
f_{17}	1.02e+02 $\pm$ 1.04e+01	1.75e+02 $\pm$ 3.72e+01	+	6.97e+03 $\pm$ 1.07e+03	+	5.14e+01 $\pm$ 2.84e-01	-
f_{18}	4.18e+02 $\pm$ 5.24e+01	1.85e+02 $\pm$ 3.40e+01	-	7.08e+03 $\pm$ 9.14e+02	+	4.94e+02 $\pm$ 1.08e+02	+
f_{19}	5.04e+00 $\pm$ 9.33e-01	4.25e+01 $\pm$ 2.66e+01	+	6.32e+00 $\pm$ 1.18e+00	+	1.40e+00 $\pm$ 2.19e-01	-
f_{20}	2.43e+01 $\pm$ 5.49e-01	2.00e+01 $\pm$ 9.04e-01	+	2.50e+01 $\pm$ 9.73e-02	+	2.28e+01 $\pm$ 7.85e-01	-
f_{21}	2.85e+02 $\pm$ 2.20e+02	9.22e+02 $\pm$ 3.06e+02	+	8.12e+02 $\pm$ 3.73e+02	+	3.27e+02 $\pm$ 2.64e+02	=
f_{22}	2.39e+03 $\pm$ 3.67e+02	3.09e+03 $\pm$ 9.98e+02	+	1.19e+04 $\pm$ 1.26e+03	+	7.58e+01 $\pm$ 8.58e+01	-
f_{23}	8.37e+03 $\pm$ 1.01e+03	8.88e+03 $\pm$ 1.20e+03	+	1.18e+04 $\pm$ 8.52e+02	+	1.05e+04 $\pm$ 1.11e+03	+
f_{24}	3.22e+02 $\pm$ 1.95e+01	2.87e+02 $\pm$ 1.47e+01	-	1.64e+03 $\pm$ 1.05e+03	+	3.56e+02 $\pm$ 9.89e+00	+
f_{25}	3.66e+02 $\pm$ 9.94e+00	3.69e+02 $\pm$ 1.78e+01	=	4.94e+02 $\pm$ 1.88e+02	+	3.96e+02 $\pm$ 1.19e+01	+
f_{26}	2.00e+02 $\pm$ 3.25e-02	3.50e+02 $\pm$ 7.93e+01	+	6.04e+02 $\pm$ 7.11e+02	+	2.09e+02 $\pm$ 3.92e+01	+
f_{27}	1.25e+03 $\pm$ 2.08e+02	1.24e+03 $\pm$ 1.56e+02	=	1.28e+03 $\pm$ 2.51e+02	=	1.79e+03 $\pm$ 8.78e+01	+
f_{28}	1.24e+03 $\pm$ 1.53e+03	4.33e+02 $\pm$ 3.26e+02	=	3.27e+03 $\pm$ 5.60e+03	=	6.33e+02 $\pm$ 8.95e+02	+